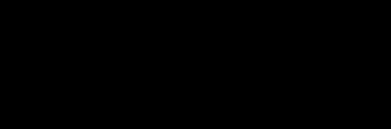


QuickTime for Java Overview

Session 408

QuickTime for Java Overview

Anant Sonone and Michael Hopkins
QTJava Engineering

QuickTime and Java

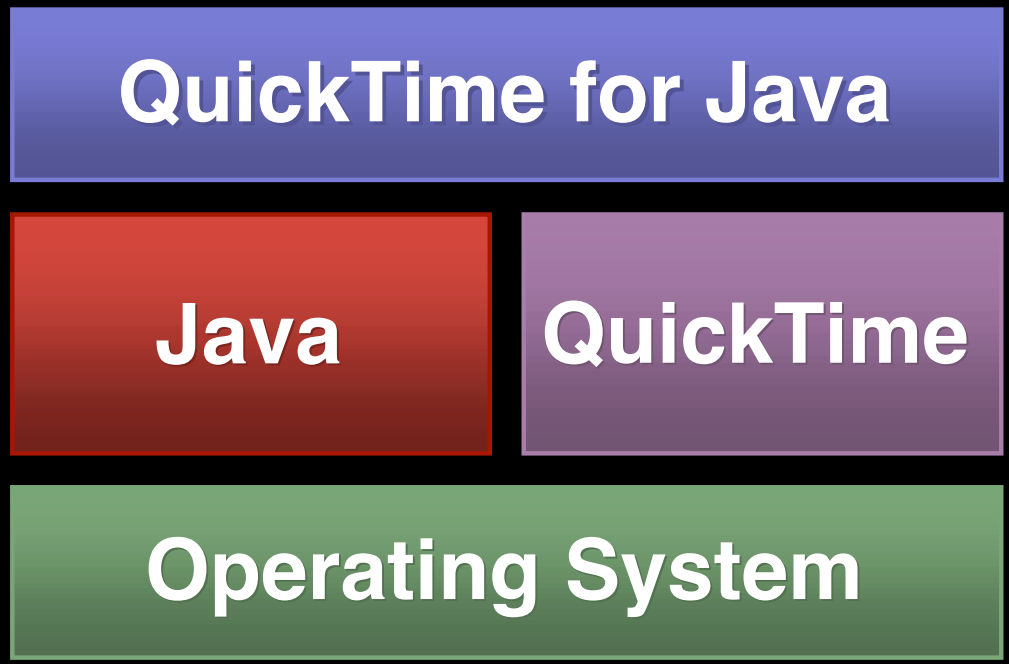
- QuickTime
  - Mature, cross-platform, flexible media architecture
    - Rich set of services supporting industry-standard media
      - Dynamic media, still images, virtual reality
- Java
  - Full-featured modern object-oriented language
    - Cross-platform deployment of applications and applets

QuickTime for Java

- QTJava is a cross-platform Java class library accessing the native QuickTime media services
  - Presents QuickTime “C” API as a set of object-oriented Java classes in logical packages
  - Provides abstracted framework of media services allowing common tasks to be done more easily

QTJava Architecture

- QuickTime for Java relies on core services from Java, QuickTime, and the Operating System

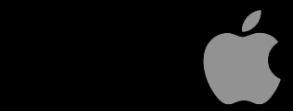


QTJava Supported Platforms

- Mac OS—MRJ 2.1 and above (JDK 1.1.8)
- Windows—Sun-compatible JRE 1.1 and above
- Mac OS X—J2SE™

QTJava in QuickTime 6

- QuickTime for Java 6 release
  - QuickTime 6 API feature support
    - Support for MPEG-4
    - Support for new Idle Manager
  - Complete coverage of QuickTime 4 and 5 API
  - Additional functionality and services
  - JDK 1.4 support (for Windows)



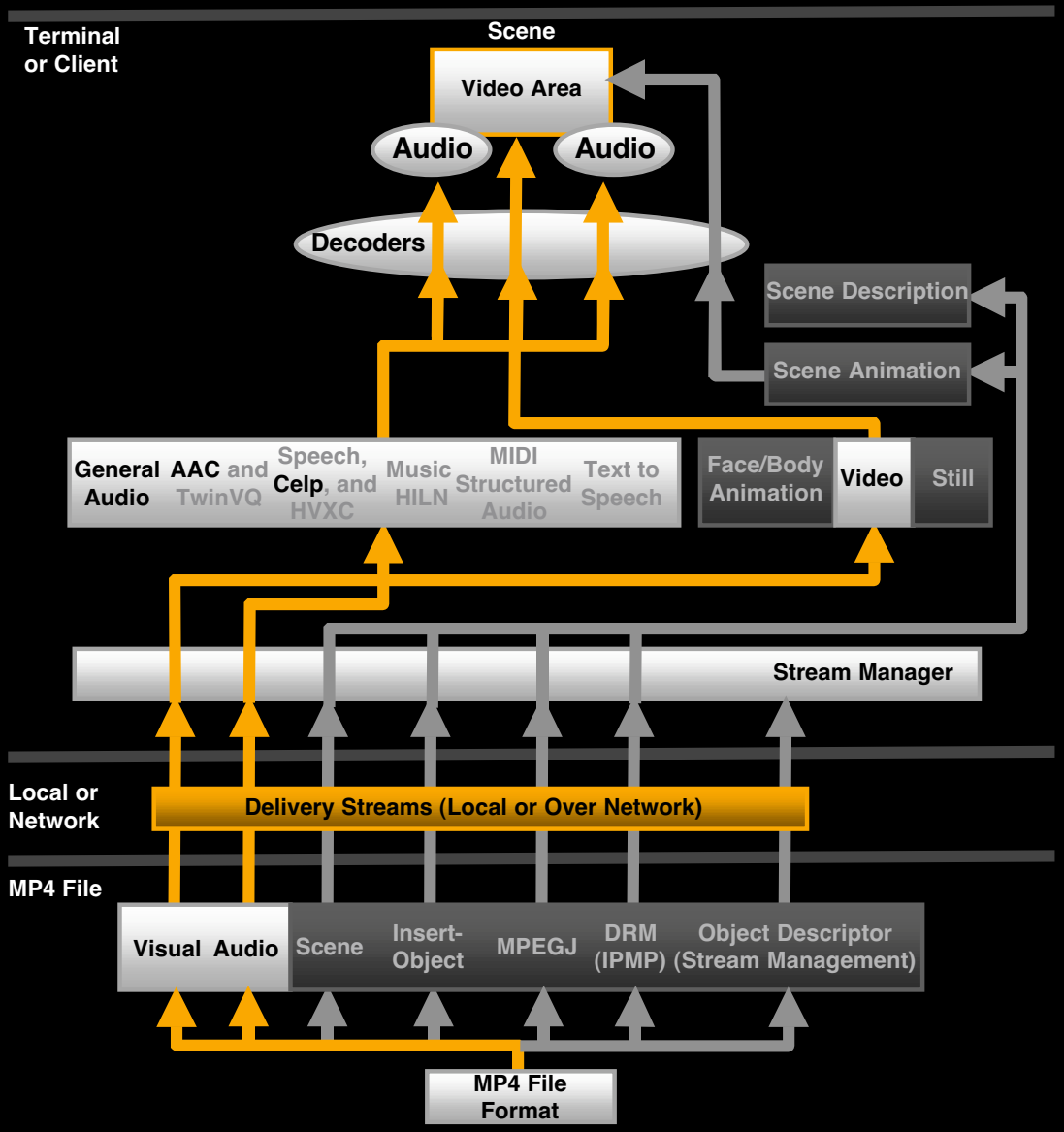
New QuickTime 6 Features

Introduction to MPEG-4

What Is MPEG-4?

- Based on the flexible component-based file format of QuickTime
- Scaleable quality based on needs of client
    - Low bandwidth for internet delivery
    - Near television quality for DSL and cable modems
- Provides support for audio and video, as well as MIDI, music, text to speech, and other technologies

MPEG-4 Architecture

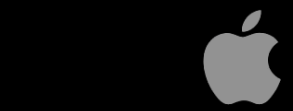


QuickTime 6 MPEG-4 Video Support

- MPEG-4 movies stored in .mp4 files
- New video codec for video compression
    - ISMA compliant
    - Conforms to Profile 0 standard
    - Low data rate of 64Kbits/second
    - Interoperable with other systems

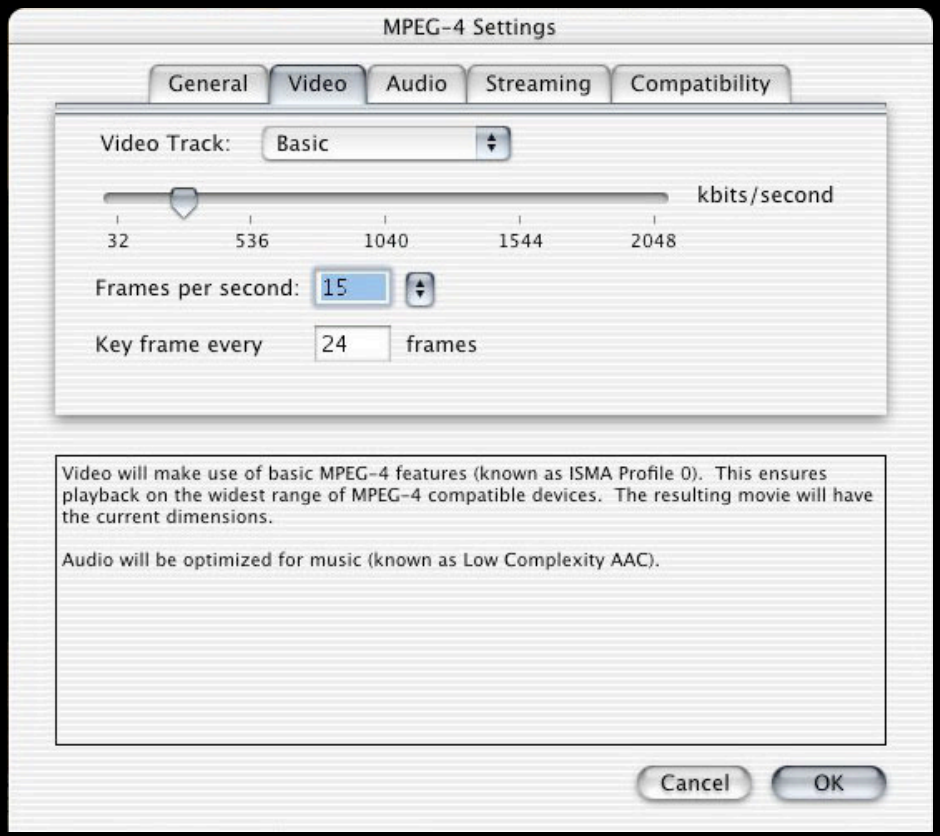
QuickTime 6 MPEG-4 Audio Support

- ISMA profile levels 0 and 1
- AAC
  - Provides high-quality, low bit rate audio
  - Superior to MPEG-3
- Celp
  - Very low bandwidth for voice and telephony
  - Language specific



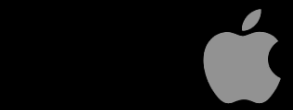
Additional MPEG-4 Features

- Support for native MPEG-4 streaming
- New dialogs for media import and export



Using MPEG-4 Features in QuickTime for Java

- MPEG-4 support is automatic!
  - No API changes
  - All applications that use QTJava can support MPEG-4 without any code changes once QuickTime 6 is installed

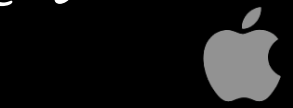


New QuickTime 6 Features

New Idle Manager API

Idle Management and Tasking in QuickTime

- Applications that use QuickTime must call routines to yield time to QuickTime
  - MCIsPlayerEvent( )
  - MCIdle( )
  - MoviesTask( )
  - TaskMovie( )
- These routines may be called implicitly by QTJava



Idle Management and Tasking Inefficiencies

- Difficult to determine how often QuickTime should be called
    - Movies may contain wired sprites that run even when movie is stopped
- Calling at periodic intervals can be wasteful
    - Often QuickTime doesn't need to be called so cycles are wasted

New Idle Management API

- Requesting idle interval

```
int delay = Movie.getTimeUntilNextTask (scale);
```

- Setting the idle interval

```
myMovie.taskAll(delay);
```

- Setting up a callback
  - Implement NextTaskNeededSooner interface and override execute method:

```
public int execute() { … }
```

New Idle Management API Callback

- Installing a callback:

  Movie.setNextTaskNeededSoonerCallback (myCallback, …)

- When QuickTime needs time, it will call the execute method of your callback class
- Removing a callback:

  Movie.removeNextTaskNeededSoonerCallback (myCallback, …)

Additional QuickTime 6 Features in QTJava

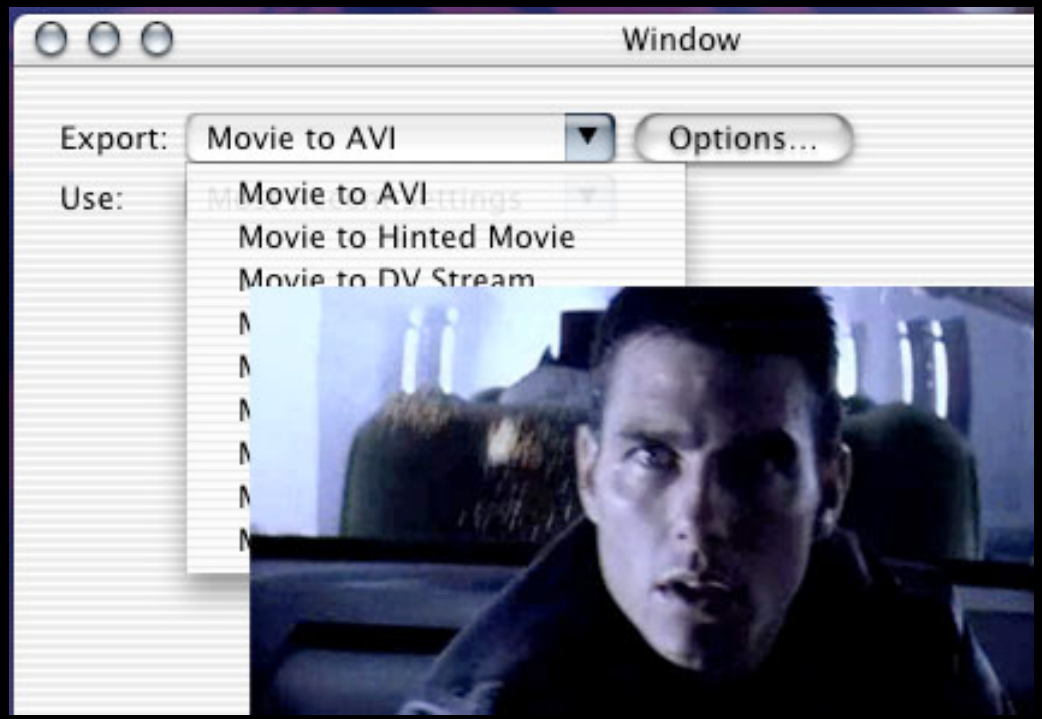
- Support for Flash 5
  - Expanded ActionScript capabilities
  - HTML text rendering
  - XML data exchange
- Support for VBR Sound Compression

New QuickTime 6 Features

Using QTJava With Swing

QTJava 6: Using QTJava With Swing

- QTCanvas is heavyweight, making use in a swing application problematic



QTJava 6: Using QTJ With Swing

- Introduced new class—JQTCanvas
  - Lightweight
  - Provides most of the functionality of QTCanvas
  - Hardware accelerated on Mac OS X
  - Easy to use

JQTCanvas Limitations

- Similar to Compositor
  - Lower frame rate than QTCanvas
  - Cannot be used to display movie controllers
  - Time-based clients must implement DrawingNotifier interface
    - SWCompositor
    - MoviePresenter
    - QTEffectPresenter

Using JQTCanvas

- Creating a JQTCanvas:

```
JQTCanvas theCanvas = new JQTCanvas
    (QTCanvas.kInitialSizeNoResize, …);
```

- Setting the client:

```
mPlayer = new MoviePlayer(mov);
theCanvas.setClient(mPlayer, true);
```

- Adding the JQTCanvas to a Frame:

```
theJFrame.getContentPane().add (theCanvas);
```

- Turning on Hardware Acceleration:

```
JQTCanvas.useMacOSXAcceleration = true;
```

Demo

Using JQTCanvas

QTJava Current Features

- Enhancements in QTJava 6
  - JDK 1.4 support (Windows only)
    - No API level changes, transparent to the user
  - New MovieMediaHandler API support
  - QTVR ViewParameter get/set calls
  - Public constructor for SGOutput
  - Allows filenames with accented characters

QTJava Current Features

- Bug fixes
  - Mac OS X dialog enhancements
  - Sequence grabber fixes
    - SGDataProc—The sequence grabber calls your data function whenever any channel component writes data to the destination movie

```
mGrabber.setDataProc(new MyProc());
class MyProc implements SGDataProc {
    public int execute (…) throws QTException {
        //where you wrote the data or zero if you didn't
          write data
            return 0; }}
```

Using the Presentation API

- New API with QT 5 (Enhanced in QT 6)
- Broadcast from a sequence grabber source
    - Audio from a device such as a microphone, CD, or DV Audio source
    - Video from a device such as a DV camera, or USB camera (CritterCam)
- Broadcast can be unicast or multicast
- Broadcast user-configurable using a settings dialog

Efficient Programming

Anant Sonone
QTJava Engineering

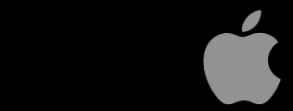
Movie Controllers

- Movie controllers present the user with an interface for controlling the movie
  - Stop/start playback
  - Jump within the timebase
  - Control sound volume
- Typically appears directly beneath the movie content

Detached Controller

When to Use a Detached Controller

- Treats movie and controller as separate graphics entities
- Allows slaving of multiple movies to a single controller
- Place movies in a Swing component
- Enables applications to use custom movie controllers



Detaching a Controller in QTJava

- Creating the canvas:

```
QTCanvas controllerCanvas = new QTCanvas();
```

- Creating the movie controller:

```
MovieController controller = new MovieController
    (movie, mcScaleMovieToFit);
```

- Detaching the controller:

```
controller.setAttached(false);
```

Using the Detached Controller (Cont.)

- Creating a QTPlayer for the controller:

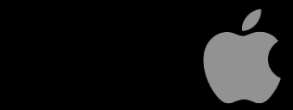
```
QTPlayer qtPlayer = new QTPlayer(controller);
```

- Setting the client:

```
controllerCanvas.setClient(qtPlayer, true);
```

- Setting up the movie:

```
MoviePlayer moviePlayer = new MoviePlayer(movie);
movieCanvas.setClient(moviePlayer, true);
```



Demo

Using a Detached Controller

Efficient Programming

Playing a Sound File

Playing a Sound File Efficiently

- Previous examples and documentation recommended using QTDrawable objects
- This methodology had many drawbacks:
    - Required use of a QTPlayer object
    - Used explicit calls to task()
    - Graphical artifacts on Mac OS X

Playing a Sound File Efficiently (Cont.)

- Open sound file as a movie:

```
Movie movie = Movie.fromFile (fileIn);
```

- Use the TaskAllMovies class to task active movies or Timebases:

```
TaskAllMovies.addMovieAndStart();
```

- Set the movie rate to 1 for playback and 0 to stop:

```
movie.setRate(1);
```

- Remove movie that no longer needs to be tasked:

```
TaskAllMovies.removeMovie();
```

Efficient Programming

QTJava Wired Sprite API

What Are Wired Sprite Movies?

- Movies that contain interactive components
- User input is translated into events which can target the movie, sprite, or track
- Events fire actions which modify individual properties of the target

Wired Sprite Movies

- Interactive movies contain sprite elements
- Sprite is typically an image that has properties that can be modified to achieve animation or interactivity
- Sprite Track defined by one or more key frame samples followed by override samples
    - Key frame contains shared image data and initial properties of the sprites
    - Override sample overrides properties of the sprites
    - Samples are based on QTAtoms

Challenges of Making Wired Sprite Movies

- QTAtom architecture is hard to understand
- Native C API is difficult and tedious to use
- Platform dependencies and endian issues
- Major development learning curve
- Third-party authoring tools may be expensive

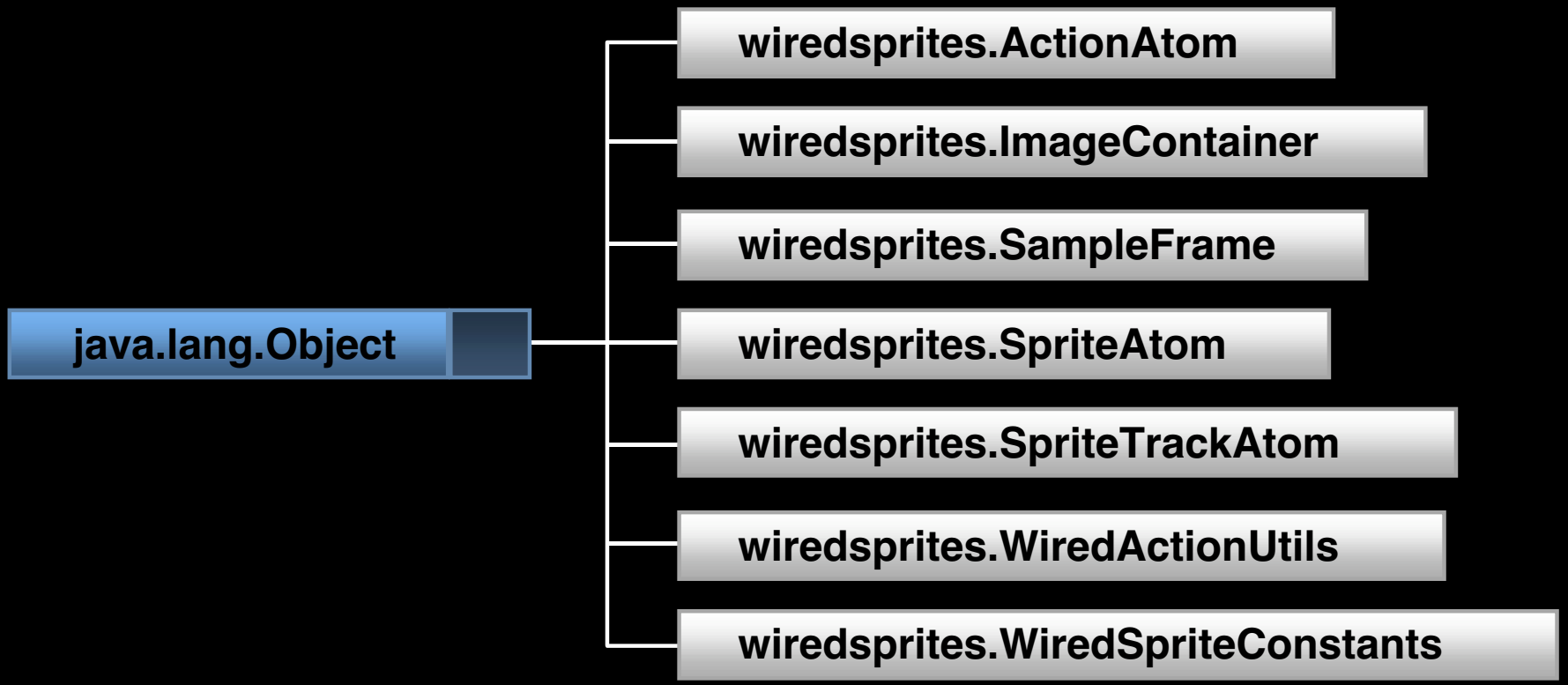
Advantages of the QTJava Wired Action API

- Easy-to-use object-oriented API
- Platform independent
- Very low overhead
- Slight learning curve

API Overview

- Creating and modifying wired movies
  - First create a new movie file with a single sprite track
  - Make key frame sample containing a sprite and all of its shared images
  - Set the sprite track's properties
  - Create override samples as needed to override matrix and image index properties of the sprite
  - Set actions on the sprites for specific user events

Wired Action Package



API Overview (Cont.)

- Creating new movie file:

```
Movie mov = Movie.createMovieFile(…);
```

- Add sprite track to movie and media to track:

```
Track spriteTrack = mov.addSpriteTrack(…);
SpriteMedia media = new SpriteMedia(spriteTrack, scale);
```

- Create key frame and add images for sprites:

```
SampleFrame keyFrame = new SampleFrame();
ImageContainer images =
        ImageContainer.makeImageContainer();
for (i=0; i < nImages; i++)
        images.addSpriteImage(…);
```

API Overview (Cont.)

- Create sprites and set their properties:

```
SpriteAtom sprite = new SpriteAtom(id);
sprite.set<Properties>(…);     // location, visibility, etc.
```

- Create actions and add to sprite:

```
ActionAtom spriteAction = new ActionAtom ()
spriteAction.set<action>(QTEvent,…);
sprite.setAction (spriteAction);
keyFrame.addSpriteAtom (sprite);
```

- Create override frame and set override properties
- Add key and override frame to media

API Overview (Cont.)

- Insert media into track:

```
spriteTrack.insertMedia(….)
```

- Create sprite track atom and set properties:

```
SpriteTrackAtom trackProperties = new
    SpriteTrackAtom (…);
```

- Add sprite track properties to the media:

```
WiredActionUtils.setTrackProperties (media,
    trackProperties);
```

- Add movie resources to the movie

Demo

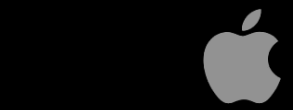
Exercising the Wired Action APIs

Efficient Programming

Installing QuickTime for Java

QTJava Standard Installation Procedure

- Mac OS X—**preinstalled!**
- Mac OS 9
    - Select Custom Install
- Windows
    - Install a Sun-compatible Java VM
    - Install QuickTime
        - Select Custom Install



QTJava Windows Custom Installation

- Licensing the Installer

  http://developer.apple.com/mkt/swl/agreements.html#QuickTime

  - Developer is responsible for insuring installation is successful
- Writing a custom installer
  - License individual pieces
  - Modify the ini file

Who to Contact

General Developer Support
Public mailing list for QTJava Developers
lists.apple.com

For General Developer Information
QTJava SDK (Sample Code, Documentation)
developer.apple.com/quicktime/qtjava

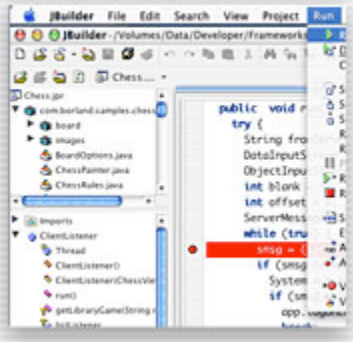
For Seeding Enrollment
Must be Registered Apple Developer
qtjava@apple.com

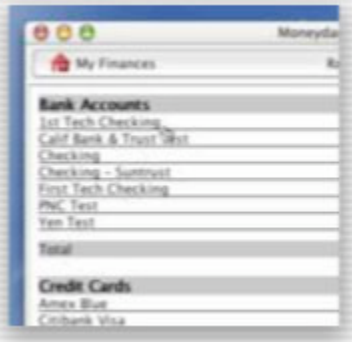
http://developer.apple.com/wwdc2002/urls.html

Roadmap

FF010 QuickTime: Let your voice be heard!	Room J1 **NOW!**
606 QuickTime for the Web: Learn about web deployment and groovy Flash media	Room A2 **Fri., 2:00pm**
607 QuickTime and MPEG-4: Technical overview of this revolutionary new media format	Room A2 **Fri., 3:30pm**

Q&A





Tom Maremaa
Apple TechPubs
maremaa@apple.com

http://developer.apple.com/wwdc2002/urls.html


WWDC2002


WWDC2002


WWDC2002